VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

2019

Oleksandr Dede

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Oleksandr Dede**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: Tieto Czech s.r.o.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Daniel Stříbný**

Konzultant bakalářské práce: Mgr. Zdeněk Dřízga

Datum zadání: 01.09.2018
Datum odevzdání: 30.04.2019

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2019 ..... Dede

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.

Tieto Czech s.r.o.
28. října 3346/91
702 00 Ostrava - Moravská Ostrava
IČO 64608041, DIČ CZ64608051

V Ostravě 29. dubna 2019

Rád bych na tomto místě poděkoval firmě Tieto Czech s.r.o. a svému manažerovi Zdeňku Dřizgovi za možnost vykonat odbornou praxi, kolegům za spolupráci a rady během praxe a vedoucímu práce Ing. Danielovi Stříbnému za věcné připomínky a podněty k bakalářské práci.

Abstrakt

Práce popisuje průběh vykonávání bakalářské práce formou individuální odborné praxe ve firmě Tieto Czech s.r.o. V průběhu praxe jsem plnil úkoly a vystupoval jsem v roli vývojáře a UI designéra. Úkolem bylo vytvořit webovou aplikaci sloužící k hledání pracovních příležitostí v rámci firmy (to obnášelo návrh uživatelského rozhraní, vývoj backendu a frontendu) a vytvořit návrhy uživatelského rozhraní pro aplikace pro veřejný sektor.

Klíčová slova: Bakalářská praxe, Java, Spring, React, UI

Abstract

This bachelor thesis describes the course of the individual professional practise in the Tieto Czech s.r.o. company. During my practice I have been fulfilling tasks and I worked as a developer and UI designer. The task was to create a staffing portal for searching of job opportunities inside of company (including designing of user interface, backend and frontend developing) and create design of user interface for civil services applications.

Key Words: Bachelor practice, Java, Spring, React, UI

Obsah

Seznam použitých zkratek a symbolů

CSS	–	Cascading Style Sheets
DB	–	Database
ES6	–	ECMAScript 6
HTML	–	Hyper Text Markup Language
JSON	–	JavaScript Object Notation
PL-SQL	–	Procedural Language for SQL, rozšíření SQL vytvořené firmou Oracle
SQL	–	Structured Query Language
UI	–	User Interface
URL	–	Uniform Resource Locator
UX	–	User eXperience

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Odbornou praxi jsem vykonával ve firmě Tieto Czech s.r.o. (dále pouze „Tieto“). Tato firma s hlavním sídlem ve Finsku je zaměřena na poskytování IT služeb. Zabývá se převážně údržbou a tvorbou informačních systémů, a to jak pro soukromý sektor, tak i pro veřejný sektor. Zákazníci pocházejí převážně ze severských zemí. Tieto vytvořilo například systém pro on-line registraci a evidenci vozidel ve Finsku nebo systém pro papírenský průmysl. Vyvíjí taktéž aplikace pro interní potřeby, aby byl zajištěn hladký chod velké mezinárodní společnosti.

V Ostravě zaměstnává přes 2 000 osob. Podle statistik Úřadu práce [1] je největším zaměstnavatelem v oboru IT a čtvrtým největším zaměstnavatelem vůbec v okrese Ostrava-město. Celosvětově Tieto zaměstnává kolem 15 000 osob, a to hlavně ve Finsku, Švédsku, České republice a Indii.

Já jsem pracoval v týmu, který vyvíjel aplikace v Javě. Můj tým byl specifický tím, že kromě stálých zaměstnanců v něm pracovala velká skupina stážistů, jejíž členové byli studenty informatických oborů na vysokých školách a já jsem byl jedním z nich. Stáž umožňuje studentům použít své znalosti v praxi a získat zkušenosti během vykonávání praktických úkolů a firmě najít nadějné budoucí zaměstnance.

Každému stážistovi byl přidělen manažerem úkol, na kterém měl pracovat. Vesměs se jednalo o tvorbu backendu a/nebo frontendu webové aplikace pro interní účely, nicméně někteří pracovali na webu pro neziskovou organizaci. Pracovalo se jak ve skupinách, tak i samostatně. Já jsem pracoval sám a mým úkolem bylo navrhnout a poté implementovat ve spolupráci s jinými kolegy webovou aplikaci sloužící jako portál pracovních příležitostí v rámci firmy a jako přehled údajů ze životopisu, takzvaný Staffing Portal. Jelikož jsem pracoval jak na backendu, tak i na frontendu, tak se dá říct, že jsem získal zkušenosti s full-stack vývojem. V rámci praxe jsem taktéž vytvářel návrhy uživatelského rozhraní (UX/UI) pro několik webových aplikací.

V následujícím textu budou podrobněji popsány mé úkoly. Nejprve rozeberu motivaci, proč bylo vůbec zapotřebí vytvářet Staffing Portal. Následně popíšu, na jaké části jsem svůj úkol rozdělil a jaké postupy a technologie jsem použil. Jedná se o následující části:

1. Návrh uživatelského rozhraní
2. Frontend – webová aplikace s využitím Reactu a Bootstrapu
3. Backend a napojení na databázi – Java aplikace s využitím frameworku Spring

Dále uvedu, jak jsem navrhoval uživatelské rozhraní pro další aplikace a na závěr vypíšu, jaké znalosti ze studia jsem využil.

2 Zadané úkoly

2.1 Staffing Portal

Cílem bylo vytvořit webovou aplikaci s názvem Staffing Portal. Jedná se o interní pracovní portál, jehož účelem je zahrnout nejpodstatnější informace z již existujících portálů na jedno přehledné místo a tímto zjednodušit práci zaměstnancům Tieta a poskytnout komfortnější uživatelský zážitek.

V současné době existuje v intranetu Tieta více aplikací související s náborovým procesem. Jedna aplikace umožňuje si vytvořit životopis a zadat do něj jak popis práce v jednotlivých pracovních anabázích, tak si vytvořit seznam schopností a jejich úrovní (mezi tyto schopnosti patří „hard skills“ jako znalosti programovacích jazyků i „soft skills“ jako schopnost dobře prezentovat, poskytovat zpětnou vazbu), které se vybírají z předpřipraveného seznamu. Další aplikace obsahuje seznam pracovních nabídek v rámci Tieta a možnost vytvořit si svůj vlastní inzerát. Tyto a další aplikace jsou vytvořeny s pomocí různých technologií a mají rozdílné uživatelské rozhraní.

Strategií Tieta je podpora takzvaných „interních rotací“. To znamená, že každý pracovník má možnost změnit pozici a vybrat si takovou, která bude vyhovovat jemu i firmě. Zájmem Tieta je co nejlépe využít lidské zdroje již přítomné ve společnosti, a proto spustilo vedení kampaň, ve které byli zaměstnanci povzbuzováni k vyplnění životopisu v rámci interní aplikace. K tomu, aby zaměstnanci se mohli snáze orientovat v takovém životopise a hledat potencionální novou pracovní pozici, bylo rozhodnuto vytvořit aplikaci Staffing.

V pilotní fázi byl kladen důraz zejména na zobrazování údajů z databází. Do budoucna se počítá s možností některé údaje editovat přímo v Staffing aplikaci a vytvořením rolí manažera a recruitera. Manažer bude mít například možnost stanovit kritéria potřebná pro přijetí do týmu.

Jako hlavní funkce aplikace v pilotní fázi byly určeny následující body:

- Zobrazení klíčových bodů životopisu každého uživatele (klíčové kompetence třízené podle kategorií, certifikáty) a kvality vyplněného životopisu.
- Zobrazení inzerátu uživatele, který indikuje zájem zaměstnance o změnu pozice v rámci Tieta.
- Zobrazení a filtrování pracovních nabídek v rámci firmy.
- Statistiky týkající se inzerovaných pracovních nabídek – například seznam nejhledanějších pozic nebo schopností.
- Možnost kontaktovat HR nebo technické oddělení přímo z aplikace.

Níže specifikuji jednotlivé dílčí úkoly.

2.1.1 Návrh uživatelského rozhraní

Mým prvním úkolem bylo vytvořit návrh uživatelského rozhraní aplikace na základě PowerPoint prezentace, kde byla popsána plánovaná funkcionalita Staffing Portalu. Tuto specifikaci jsem obdržel od svého nadřízeného, který ji pro změnu obdržel od zadavatele. Úkol byl přidělen našemu týmu, protože jakožto vývojáři interních aplikací pomáháme zjednodušovat práci týmům v oddělení administrativy.

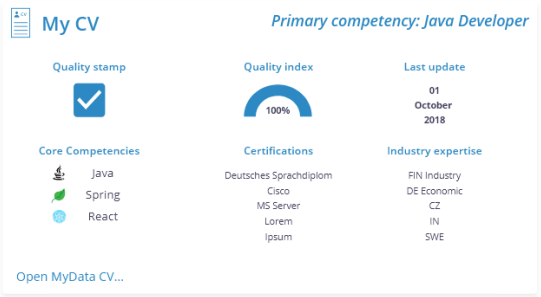
Návrh aplikace měl zahrnovat následující:

- Detaily o vyplněném životopisu – informaci, zda byl schválen nadřízeným, index kvality, datum poslední editace, seznam nejdůležitějších kompetencí, certifikátů a zkušeností v průmyslu
- Detaily o zveřejněném inzerátu – možné datum nástupu, hledaná pracovní pozice, jaký podíl z pracovní doby může zaměstnanec věnovat danému projektu
- Seznam dostupných pracovních nabídek a základní informace o nich (název pozice, datum nástupu, lokace, požadovaný podíl z pracovní doby)
- Statistiky týkající se pracovních nabídek – 3 nejžádanější pracovní pozice a schopnosti
- Nastavení aplikace (zobrazování fotografie v exportovaném životopise, zařazení detailů ze životopisu do anonymních statistik)
- Statistiky o počtu exportů životopisu

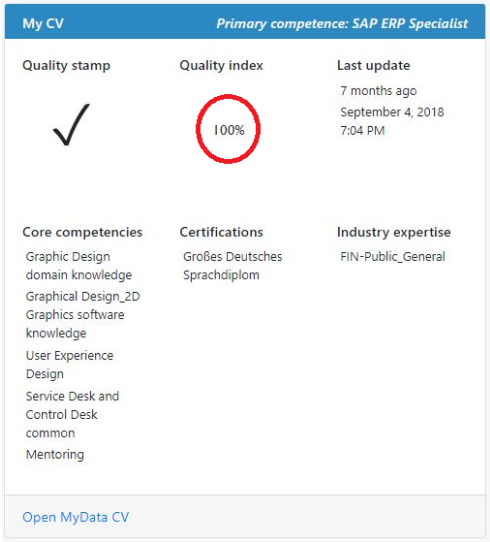
Jelikož jsem neměl k dispozici další pokyny, způsob, jak zpracovat návrh, byl v mé kompetenci. Jako inspiraci jsem použil již existující weby a příručky o webdesignu. Pro tvorbu návrhů jsem použil software Adobe XD od společnosti Adobe. Mou snahou bylo, aby návrh byl přehledný a přívětivý pro uživatele. Při tvorbě uživatelského rozhraní jsem se snažil ctít korporátní barvy a styl a také se řídit aktuálními trendy v oblasti webdesignu. Zvolil jsem třízení funkcionalit do jednotlivých záložek (viz obrázek 1) za účelem jejich snadného rozlišení.

Obrázek 1: Třízení funkcionalit do záložek

Během první konzultace se zadavatelem jsem ale zjistil, že přesně takovému řešení se on chtěl vyhnout, jelikož si přál mít všechny informace na jedné stránce. Na základě tohoto faktu jsem usoudil, že jeho požadavku bude odpovídat tzv. dashboard (z anglického překladu „nástěnka“, má za cíl zobrazovat klíčové informace na jednom místě a v jednotném stylu) [2] a podle toho



Obrázek 2: Původní návrh widgetu



Obrázek 3: Nový návrh widgetu

jsem vytvářel následující verze návrhu aplikace, protože řešení aplikace v podobě dashboardu mi už zadavatel schválil.

Místo do záložek jsem umístil jednotlivé funkcionality do menších oken, takzvaných widgetů, které se kromě v dashboardech vyskytují hlavně na mobilních zařízeních. Widgety bylo zapotřebí nějakým způsobem od sebe odlišit. Jelikož je pro lidské oko snazší číst text na bílém pozadí [3], rozhodl jsem se jednotlivé widgety navrhnout v bílé barvě a ohraničit je pomocí stínování. Mou snahou bylo, aby se všechny widgety vešly na obrazovku takovým způsobem, aby nebylo nutné posouvat stránku a zároveň aby jednotlivé elementy nebyly příliš malé, nepřehledné, a nebo nahuštěné příliš blízko k sobě.

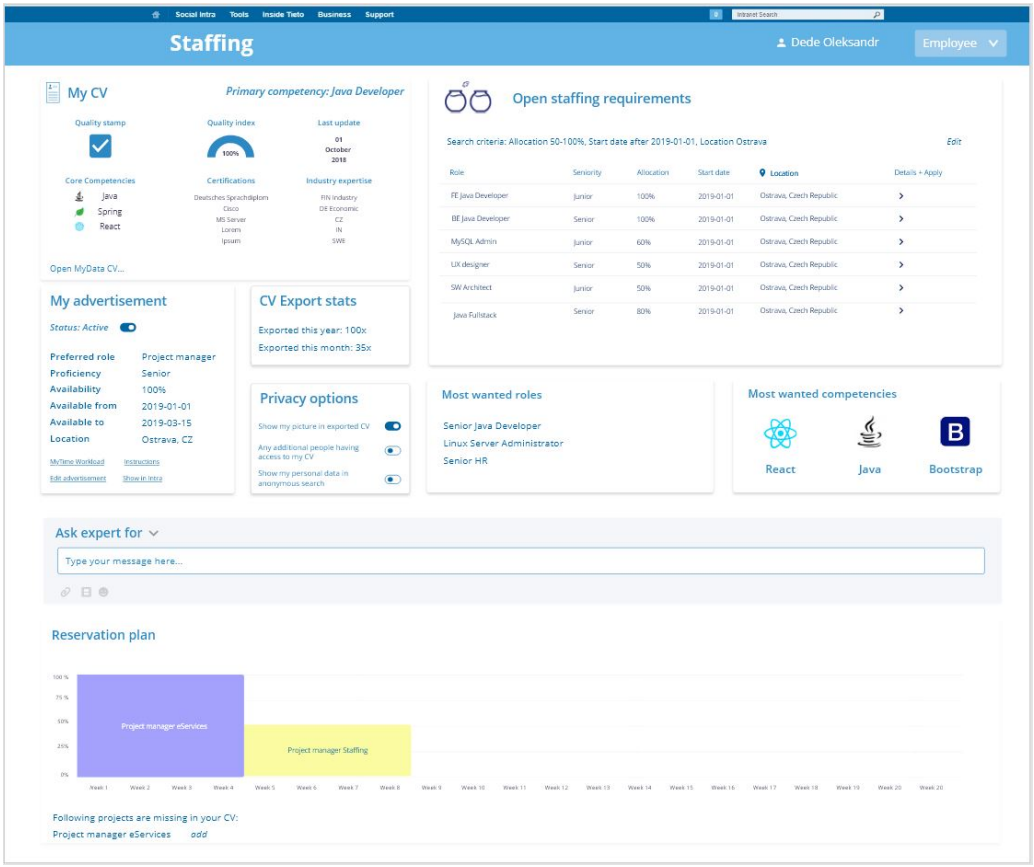
To se mi podařilo, nicméně zadavatel se později rozhodl, že do aplikace bude nutné přidat další prvky, a to okno umožňující psaní dotazů na oddělení HR nebo na technické oddělení a časový harmonogram ukazující dostupnost zaměstnance v jednotlivých týdnech a jeho obsazenost na jednotlivých projektech. Tyto dva prvky jsem přidal pod stávající elementy. Se zadavatelem jsem dolaďoval i další detaily týkající se jednotlivých funkcionalit, například rozložení detailů ve widgetech a přidání dalších informací jako primární kompetence. Nechal jsem ho vybrat mezi dvěma variantami rozložení widgetů. Poté, co zadavatel schválil finální návrh (viz obrázek 4), začal vývoj frontendu.

Jelikož došlo během vývoje frontendu ke změně designu intrawebu a s ní se změnila barva globální navigační lišty na bílou, bylo nutné přistoupit k menší změně vzhledu uživatelského rozhraní. Abych nepoužíval tři rozdílné styly na jedné stránce (pro globální navigaci, lokální navigaci a samotný obsah aplikace), rozhodl jsem se sjednotit styl lokální a globální navigace. To znamenalo změnit barvu lokální navigace z modré na bílou. Nicméně tímto došlo k vymazání rozdílu mezi navigací a obsahem navigace. Z tohoto důvodu jsem jako pozadí obsahu navigace použil tmavší odstín bílé barvy. Vedlejším pozitivním účinkem bylo to, že se barevně odlišily

widgety od pozadí aplikace, a to v přijatelné míře.

Aby aplikace nebyla jen změtí bílých elementů pod bílou navigací, pozměnil jsem design widgetů. Vytvořil jsem hlavičku, která byla na rozdíl od zbytku widgetu v modré barvě a obsahovala název widgetu. Widget My CV obsahoval v hlavičce i název primární kompetence a měl také patičku, který obsahoval odkaz na web, ve kterém je možné upravit životopis (viz srovnání obrázků 2 a 3).

Nakonec ani po úpravách nebyl návrh uživatelského rozhraní konečný. V průběhu implementace backendu a frontendu došlo k několika změnám. Například bylo rozhodnuto, že kalendář ukazující vytížení zaměstnance bude vytvořen jiným týmem a bude pouze importován z externího zdroje. Také z pilotní fáze byla vyřazena statistika exportů životopisu, protože ta není prozatím zaznamenávána do globální firemní databáze.



Obrázek 4: Návrh uživatelského rozhraní

2.1.2 Vývoj frontendu

Dále následovalo převedení grafického návrhu do kódu. Frontend jsem se rozhodl udělat jako React aplikaci, protože na Reactu jsou postaveny i jiné aplikace udržované mým týmem a je jednodušší spravovat aplikace, pokud byly vytvořeny podobným způsobem.

React je open source javascriptová knihovna pro vytváření uživatelských rozhraní vytvořená a udržovaná Facebookem. Umožňuje vytvářet komplexní UI z malých a oddělených kusů kódů, kterým se říká „komponenty“. [4] Ty používáme, aby React aplikace věděla, co má zobrazit na obrazovce. Pokud se změní data, React zaktualizuje a znovu načte komponenty. React je napsán tak, aby bylo možné ho kombinovat se širokou škálou jiných javascriptových knihoven a přizpůsobit tak aplikace našim potřebám.

Jelikož React samotný neobsahuje správu stavů aplikace, použil jsem taktéž knihovnu Redux, která tento problém řeší. Redux je stavový kontejner pro javascriptové aplikace, jinými slovy pomáhá nám spravovat data, která zobrazujeme a určit způsob, jak reagovat na vstupy uživatele. [5] Redux se řídí 3 základními principy:

- **Jeden zdroj pravdy** - Stav celé aplikace je uložen v jednom store (úložišti) To ukládá do stromové struktury historii stavů.
- **Stav lze jen číst** - Pokud se řídíme principem Immutable, tedy neměnností daných objektů, tak místo editace stávajícího stavu se vytvoří nový.
- **Změny se provádějí jednoduchými funkcemi** Takové funkce se nazývají reducery. Vezmou předchozí stav a akci a vrátí následující stav. [6]

React+Redux aplikace má v našem případě následující strukturu:

public

Obsahuje veřejnou část aplikace, například index.html nebo ikonu aplikace.

src

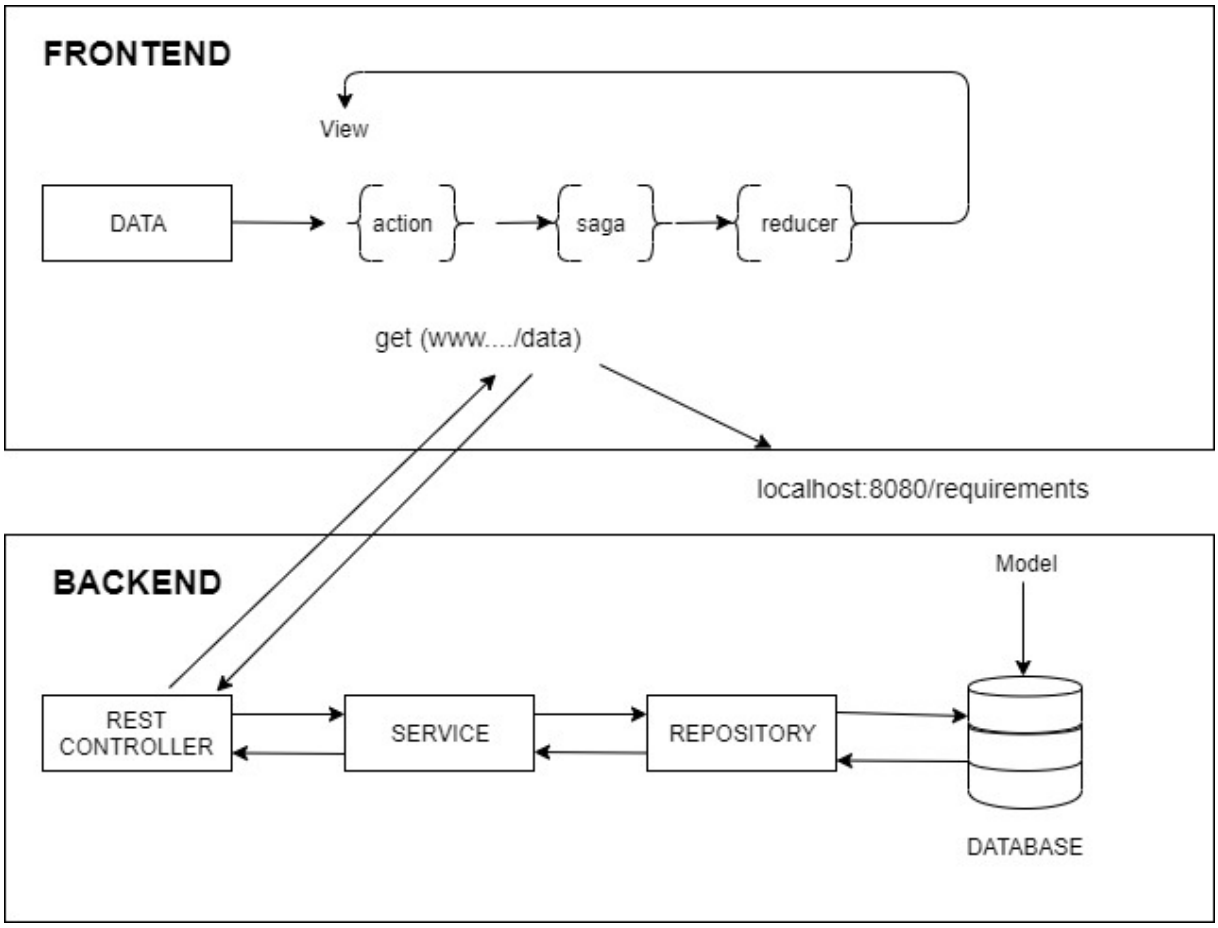
Složka src (z anglického source) obsahuje neveřejnou část aplikace, tedy zdrojové kódy obsahující logiku aplikace. Obsahuje níže uvedené složky:

- Components – Zde najdeme vícekrát používané komponenty. Obzvlášť u velkých projektů používání komponent ušetří práci a taky nám pomohou se vyhnout tzv. špagetovému kódu. Takovéto oddělování znovupoužitelných částí není nutné pro běh programu, nicméně činí kód přehlednějším. V našem případě nemají komponenty velké opodstatnění, protože drtivá většina kódu widgetů se neopakuje.
- Containers – Zpravidla se jeden kontejner rovná obsahu jedné stránky. Kromě index.js souborů obsahují i actions.js, constants.js, reducer.js a sagas.js soubory.
  - actions.js - Indikuje záměr změnit stav. V našem případě se jedná o záměr získat data z jednotlivých JSON souborů. Akce jsou jedinou cestou, jak dostat data do store. [7] Definuje funkce (a případně jejich vstupní parametry), které budeme využívat. V tomto případě se jedná o funkce pro získávání (fetchování) uživatelských dat z jednotlivých JSON souborů, které produkuje backend.

  - constants.js - Zde definujeme konstanty pro reducer, symbolizující stavy aplikace. Nejsou nutné pro běh programu, ale jejich definici ve vyhrazeném souboru doporučuje autor Reduxu Dan Abramov, [8] protože díky constants.js je snadné udržovat a editovat typy akcí.
  - index.js - Jádro kontejneru. V něm se kombinuje JavaScript a HTML. Zatímco v Javascriptu definujeme proměnné, konstanty a funkce, HTML kod dáváme zpravidla do funkce return a v něm definujeme grafický vzhled aplikace a dosazujeme do něj parametry. Volají se v něm actions a získávají data z reduceru.
  - reducer.js - Přímo navazuje na backend. Obstarává state aplikace resp. kontejneru. Jeho úkolem je namapovat data do jednotlivých stavů. Do jednotlivých stavů můžeme zahrnout více namapovaných položek. Také můžeme nastavit reakci pro případ, že se fetchování dat nepodaří. Mezi jednotlivými fetchováními přepínáme pomocí switche.
  - sagas.js - Zjednodušuje asynchronní komunikaci. Do sagy zapisujeme URL koncových bodů (endpointů), ze kterých získáváme data.

- Global - actions.js, constants.js - soubory ovlivňující všechny kontejnery
- Internals - generators, server, styles - obsahuje ES6, základ pro JavaScript (tyto dva pojmy jsou často zaměňovány), knihovnu Bootstrap a můj vlastní soubor s CSS styly určenými výhradně pro Staffing Portal
- Media - Zde najdeme obrázky použité v aplikaci.
- Models – Navazují na model na backendu. Definují jednotlivé položky JSONu.

Níže bude vysvětleno na konkrétním příkladu, na jménu zaměstnance, jak probíhá získání a zpracování paramteru z backendu.



Obrázek 5: Schéma funkcionality backendu a frontendu a jejich propojení

Nejprve jsem si musel definovat model databázové tabulky, tak aby odpovídal provedení na backendu.

```
// Model/Employee.js
export const cEmployee = {

    displayName: 'displayName',
    (...) // další parametry
    }

export const Employee = new Map({
    [cEmployee.displayName]: '',
    (...)
    }

export const mapEmployee = src =>
    new Map({
   [cEmployee.displayName]: src[cEmployee.displayName],
});
```

Výpis 1: Model databázové tabulky na frontendu

Dále bylo třeba v souboru actions.js definovat funkci, která bude získávat data. Jelikož záleží na přihlášeném uživateli, funkce má vstupní parametr id uživatele. Funkci jsem zaznamenal i v souboru constants.js. Každá funkce odpovídá jednomu JSONu.

```
export function fetchEmployee(id) {
 return {
   type: FETCH_EMPLOYEES,
   id: id
 };
}
```

Výpis 2: Actions.js - funkce na fetchování JSONu

V reduceru namapujeme jednotlivé JSONy, případně jejich vnořené části, a procházíme jednotlivé stavy pomocí switche. Náš vybraný parametr je načten pomocí mapEmployee.

```
// reducer.js
case `${FETCH_EMPLOYEES}${FULFILLED}`:
        const getEmployee = mapEmployee(action.response);
        // (... načteme další mappery, které jsou součástí FETCH_EMPLOYEES...)
        return state.setIn([c.employeeLoading], false)
             .setIn([c.employee], getEmployee)
        // (...)
```

Výpis 3: Reducer.js - mapování JSONů

V souboru saga.js jsou definované endpointy.

```
// saga.js
export function* getUserSaga(action){
        try {
            console.log("Get user Saga called", '/employee/' + action.id.
                toString());
            const response = yield call(request, url(`employee/${action.id}`),
                get());
            console.log("Get user response: ", response);
            yield put(fulfilled(FETCH_EMPLOYEES, response));
        } catch(err) {
            console.log("Get user Saga failed");
            yield put(rejected(FETCH_EMPLOYEES, err));
        }
    }
```

Výpis 4: Saga.js - definice endpointů

Pokud vše proběhne v pořádku, tak se nám v aplikaci zobrazí jméno přihlášeného uživatele.

Výhodou JavaScriptu je početná komunita, která vytváří open source knihovny a doplňky. Dvě takové knihovny jsem využil.

První z nich je moment.js, díky kterému lze snadno zpracovávat výstupy týkající se časových údajů a ty prezentovat v uživatelsky přívětivější podobě. Například zobrazuji nejen datum poslední editace životopisu (například 12.3.2019), ale také dobu, která uplynula od editace (například před měsícem).

Druhou knihovnou je Victory.js, která usnadňuje zobrazovat číselné výstupy v grafech.

Níže následuje popis vývoje backendu, který přiblíží tvorbu JSON souborů, které jsme využívali na frontendu.

2.1.3 Vývoj backendu a práce s databází

Backend byl napsán v Javě s pomocí frameworku Spring. Tento framework se stará o infrastrukturu kolem aplikace, díky čemuž se můžeme soustředit na vlastní aplikaci.

Díky Springu nebylo nutné psát konstruktory, kde jsou zapotřebí všechny parametry nebo žádný parametr. Stačilo pouze uvést anotace @AllArgsConstructor, respektive @NoArgsConstructor.

Spring také zajišťuje propojení těch tříd, které definují tabulky v databázi. K tomu slouží anotace @OneToOne, @OneToMany, @ManyToOne a @ManyToMany. Anotace @Entity značí, že daná třída symbolizuje tabulku v databázi. Anotací @JsonIgnore můžeme z výsledného JSONu vyřadit nepotřené hodnoty.

Struktura Springové aplikace je následující:

- **StaffingApplication**

  Zde spouštíme aplikaci. Také tam najdeme testovací konzolové výpisy, které usnadňují hledání chyb.

- **application.properties**

  Slouží k nastavení připojení k databázi a práce s jejím obsahem.

- **Model** Třídy ve složce Model odpovídají struktuře datového modelu. Tyto třídy v kombinaci s anotacemi značícími vazby mohou automaticky vytvořit datový model. Této vlastnosti jsem ale využil jen při testech, kdy jsem aplikaci testoval na testovací databázi. V provozní verzi jsou data získávána z globální firemní Oracle databáze.

Níže je ukázka jedné třídy modelu, kde můžeme vidět automatickou tvorbu konstruktorů pomocí anotací, definici parametrů a způsob, kterým jsou vytvářeny mezitabulkové vazby.

```
@Entity
@Table(name = "EMPLOYEES")
@Getter
@Setter
@NoArgsConstructor
public class Employee {


    @Id
    @Column(name = "ID")
    @JsonIgnore private Long personID;

    @Column(name = "DISPLAY_NAME")
    private String displayName;

      (...) //definice dalších parametrů


   @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, mappedBy
       = "employeeID")
   @JsonIgnore
   private Set<EmployeeCompetences> employeeCompetencies;

   (...) //definice dalších mezitabulkových vazeb
```

Výpis 5: Model DB tabulky na frontendu

- **Repository**

  Interface, kde se zadávaly SQL příkazy na získání dat z databáze v případě, že bylo třeba použít jiné SQL dotazy, než ty, které jsou definovány defaultně ve Springu (obecně vzato se jedná o CRUD operace). Zde uvedená funkce získává seznam pěti kompetencí zaměstnance, které nepatří k průmyslovým zkušenostem, certifikátům nebo jazykům.

```
@Query(value = "SELECT segment3 " +
        "FROM PERSON_COMPETENCES " +
        "WHERE employee_id = :employeeID AND segment1 != 'Industry' AND
           segment2 !=     'Certificates' AND segment2 != 'Language'" +
        "ORDER BY level_value DESC " +
        "FETCH FIRST 5 ROWS ONLY;",
        nativeQuery = true)
List<String> getCoreCompetencies(@Param("employeeID") Long employeeID);
```

Výpis 6: Repository.java - definice SQL dotazu

- **Service + Service/impl**

  Obsahuje volání funkcí získávající data z databáze, a to jak pomocí defaultních funkcí definovaných přímo ve Springu, tak i pomocí funkcích, které jsou definovány ve složce Repository. Z ukázky kódu lze vidět, jak se volá funkce na volání kompetencí, která je uvedena výše v sekci o složce Repository.

```
List<String> getCertifications(Long personID);


@Autowired
private EmployeeRepository employeeRepository;

//parametr PersonID určuje, údaje jakého zaměstnance mají být zobrazeny
    @Override
    public List<String> getCoreCompetencies(Long personID) {
        return employeeRepository.getCoreCompetencies(personID);
    }
}

(...) //Další funkce na získání dat
```

Výpis 7: Service.java - volání funkcí pro získávání dat

- **Controller**

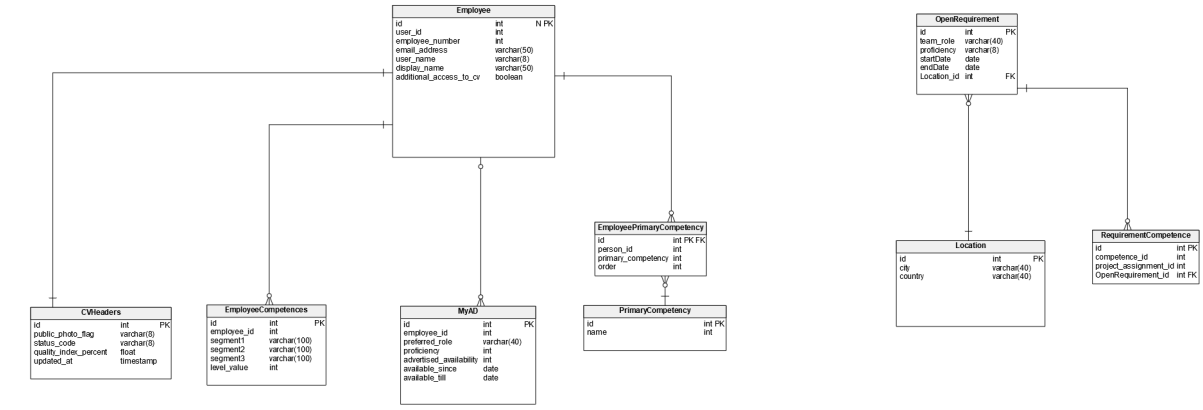
  Převádí data do backendu. V mém případě generuje JSON. Definujeme zde URL backendu a také jednotlivé odkazy.

```java
@RestController          //aplikace je připravena pro použití Springem MVC
    k práci s webovými requesty (žádostmi)
@RequestMapping("/employee")
@CrossOrigin(origins = "http://localhost:3000")
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;

    @GetMapping("/{personID}")
    public Employee findById(@PathVariable Long personID) {
        return employeeService.findById(personID);
    }

    @GetMapping("/{personID}/corecompetencies")

    @RequestMapping("/{personID}/corecompetencies")

    public List<String> getCoreCompetencies(@PathVariable Long personID) {
        return employeeService.getCoreCompetencies(personID);
    }
```

Výpis 8: Controller.java

Tvorba backendu obnášela nejen tvorbu Spring aplikace, ale také práci s firemní databází. Jelikož data ve firemní databázi nejsou uspořádána stejným způsobem, kterým jsou zobrazována na frontendu, bylo zapotřebí prozkoumat větší množství tabulek v globální databázi a hledat paramtery, které je třeba zobrazit v aplikaci. S tímto vyhledáváním mi pomáhali kolegové, kteří měli větší zkušenosti s používáním této globální databáze, protože orientace v databázi, která obsahuje mnoho údajů a stovky tabulek, včetně těch, která vůbec nesouvisejí se Staffing Portalem, je poměrně obtížná.

Poté, co jsem v seznamu tabulek našel ty, které byly potřeba pro Staffing Portal, bylo zažádáno o vytvoření nového databázového uživatele. Takový účet má k dispozici zrcadlené tabulky, nutné k běhu aplikace. Přes tento účet backendová aplikace získává data.



Obrázek 6: Databázový model

2.2 Návrh uživatelského rozhraní pro další aplikace

Mým druhým velkým úkolem, který s tím prvním, tvorbou Staffing Portalu, souvisel spíše okrajově, byla tvorba návrhů uživatelského rozhraní pro sadu aplikací zvanou eServices. Jejím účelem je poskytnout městským samosprávám nebo vládním organizacím možnost provádět administrativní úkony a služby pro občany (například zaplacení poplatku za odpad, registrace nového vozidla) přes internet.

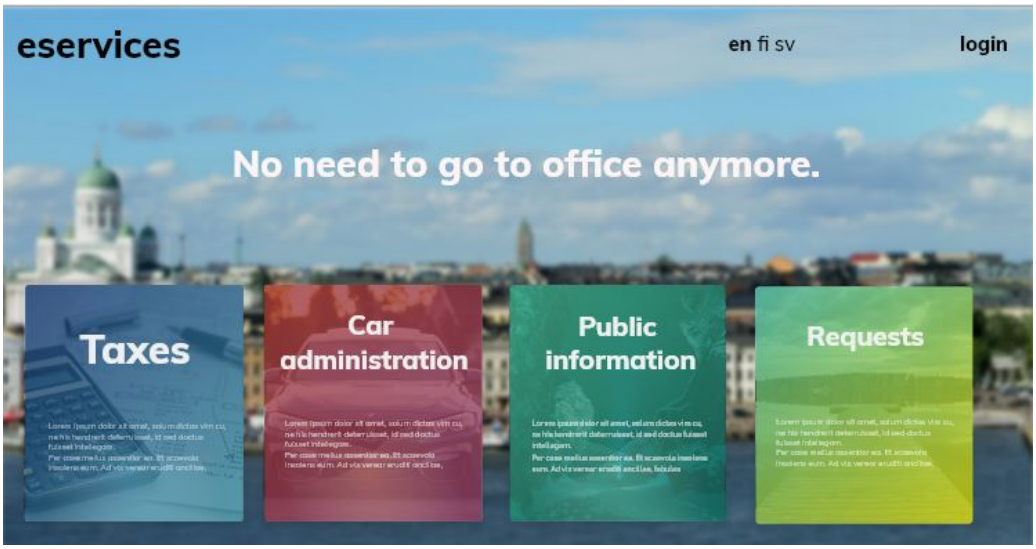
V tomto projektu jsem měl poněkud jinou roli. Měl jsem spolupracovat se zadavatelem, který zároveň byl vývojářem a tedy určoval pracovní postup a technologie, a také s programátory, kteří implementovali aplikaci a řídili se mými návrhy, které schválil zadavatel.

Obecně vzato, můj pracovní postup probíhal následovně: Zadavatel mě seznámil s vizí programu a jeho účelem. Také mi vysvětlil, jak funguje infrastruktura již existující části programu, abych tomu přizpůsobil své návrhy. Každý týden probíhaly meetingy, kde jsem já i vývojáři prezentovali svůj pokrok. Zadavatel se ujistil, zda můj návrh odpovídá požadavkům a zadal mi i programátorům další úkoly. Jako vzor pro UI komponenty byl schválen Material Design od Googlu a Bootstrap od Twitteru. Programátoři se mnou probírali, jestli je možné v rozumném čase implementovat mé návrhy a zda správně pochopili můj myšlenkový postup, případně jsme společně řešili změny v grafickém návrhu.

V následujících odstavcích bude podrobněji vysvětlena má pracovní náplň.

V první fázi jsem měl navrhnout hlavní menu pro sadu aplikací eServices. Původně jsem chtěl tlačítka odlišit obrázky, které by byly zastíněné gradientovým vzorem pro lepší čitelnost textu (viz obrázek 7), nicméně jsem později od toho upustil, protože takový způsob odlišení tlačítek jen ztěžuje orientaci uživateli a také nezanechá dobrý estetický dojem.

Kvůli tomu jsem se rozhodl použít pro tlačítka Material Icons od Googlu. Při výběru sady ikon jsem se řídil i licenčními podmínkami, aby nedošlo k jejich narušení a k finančním postihům. Material Icons jsou navrženy pro použití jak na desktopových, tak na mobilních operačních systémech.



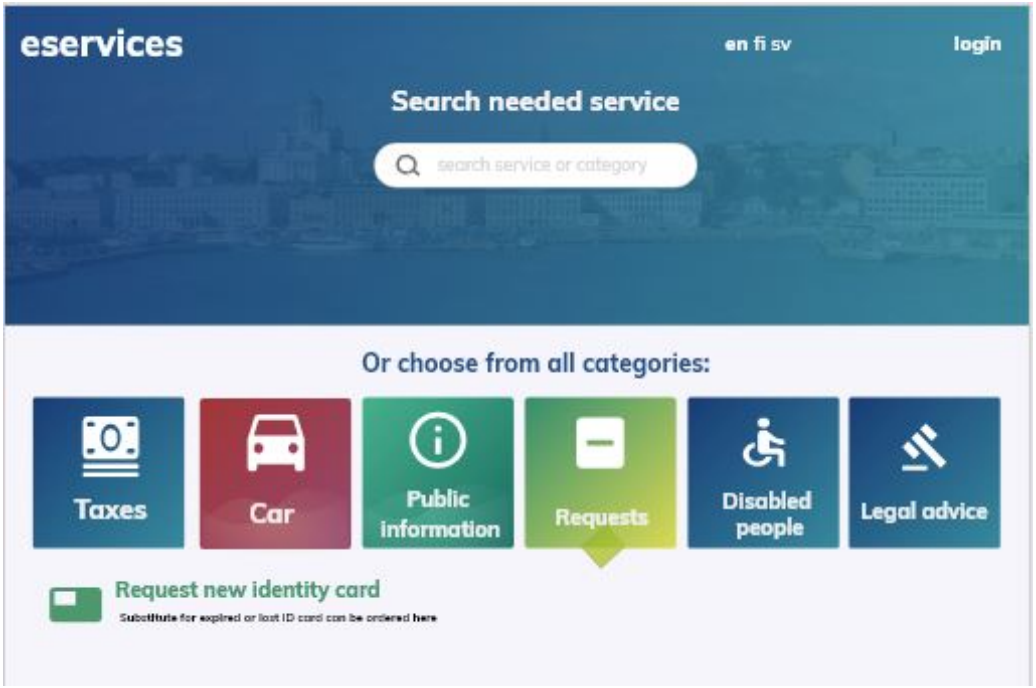
Obrázek 7: První návrh hlavního menu pro uživatelskou část eServices

Používání jednoduchých ikon a jednoduchých prostředí vůbec je trend, který lze v poslední době pozorovat ve světě grafického designu a zasáhl i do sféry vývoje softwaru. Jelikož počítačové programy a mobilní aplikace z velké části používají lidé, kteří nerozumí tomu, na jakém principu software funguje v pozadí (o to více to platí u aplikací, které jsou určeny pro lidi všech věkových kategorií, jak tomu je v tomto případě), tak je velmi důležité navrhovat aplikace tak, aby byly „user friendly“ (uživatelsky přívětivé). Pokud by totiž byly aplikace náročně na používání, snaha o účelný a přehledný kód by přišla vniveč.

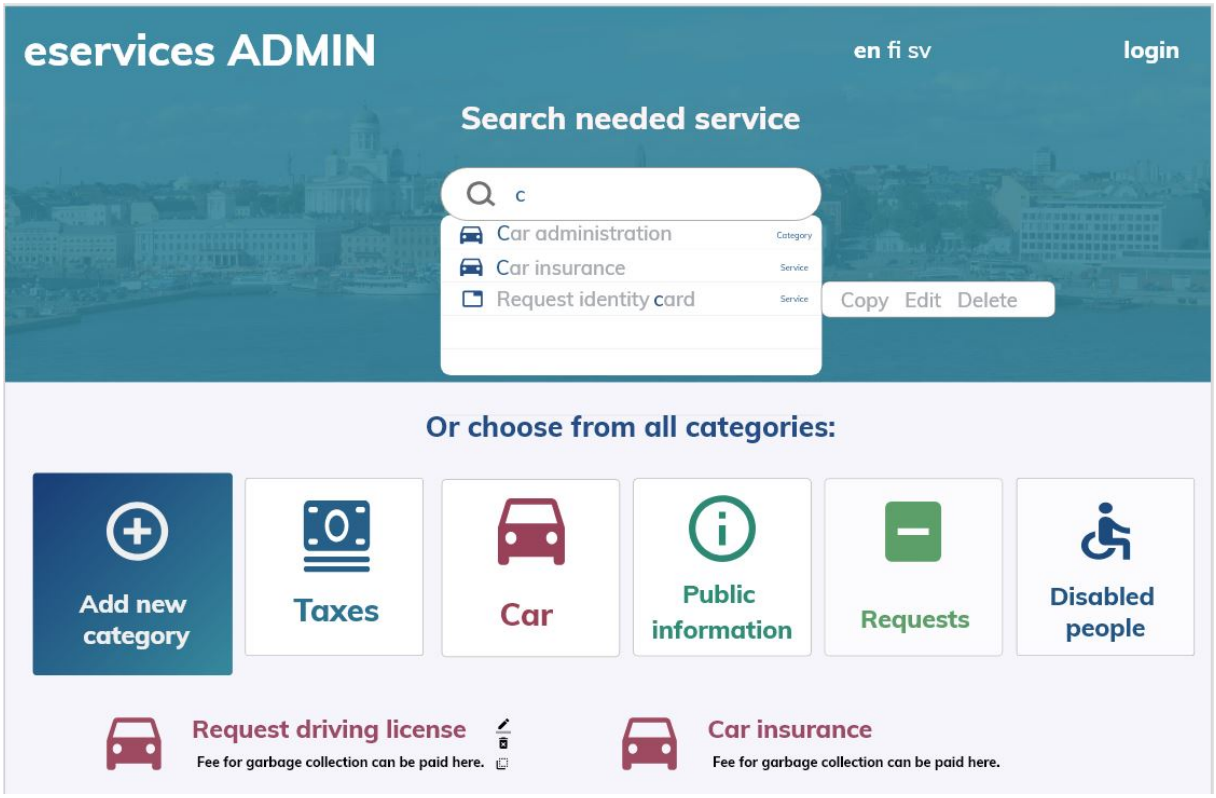
Při návrhu UI jsem řešil nejen tlačítka, ale také způsob vyhledávání (viz obrázek 8).

Vzhledem k tomu, že ne každý potencionální subjekt, který by si pořídil eServices, by využil celou uživatelskou část aplikace (například tehdy, když už obdobu eServices má nasazenou), byl v pozdější fázi kladen důraz spíše na návrh administrátorskou část eServices (viz obrázek 9). Na můj podnět se zadavatel rozhodl, že administrátorská část má být podobná té uživatelské, aby pro administrátora (který bude často mimo řady techniků) bylo snadné udržovat systém bez pomoci vývojářů.

Posledním úkolem týkající se eServices bylo navrhnout prostředí pro tzv. CodeSet Editor. Účelem editoru je editace číselníků (např. seznam druhů poplatků, států,...) bez zásahu programátora.



Obrázek 8: Vylepšený návrh hlavního menu pro uživatelskou část eServices



Obrázek 9: Návrh hlavního menu pro administrátorskou část eServices

3 Závěr

Vytvořil jsem základní verzi požadovaného pracovního portálu, který jsem kvůli časové náročnosti a komplexnosti zadání nestihl dokončit do konce. Nicméně systém je vyvinut dostatečně na to, aby mohl být v krátké době zprovozněn.

Během své praxe jsem využil znalostí získaných během studia i těch získaných během své praxe. Používal jsem jazyky Java, SQL a JavaScript. Taktéž jsem použil HTML a CSS pro kódování. Využil jsem znalosti hlavně z následujících předmětů:

- **Programování I**
- **Programování II**
- **Programovací jazyky I** – V tomto předmětu jsem získal znalosti programovacího jazyka Java. Naučil jsem se jeho syntaxi a také jeho využití v kontextu objektově orientovaného programování.
- **Úvod do databázových systémů** – Tento předmět mě naučil obecné principy relačních databází.
- **Databázové a informační systémy** – Zde jsem rozšířil znalosti z UDBS a naučil se pracovat s procedurálními rozšířeními.
- **Uživatelské rozhraní** - Díky tomuto předmětu jsem mohl lépe navrhovat uživatelská rozhraní, která jsou jednoduchá pro práci.

Co mi naopak chybělo, byla praxe s prací na frontendu obecně (předměty v prvních semestrech byly zaměřeny spíše na backend) a frameworky jako Spring a React. Nicméně bez znalostí základného programování a algoritmů bych nemohl se učit složitější technologie.

Kromě získání technických znalostí a jejich uplatnění v praxi jsem si vylepšil i soft skills - schopnost řešit náročné situace, umět se domluvit s kolegy a rozvrhnout si čas na jednotlivé úkoly.

Co se týká časové náročnosti úkolů, tak mohu říct, že každá stěžejní část (tedy backend, frontend a návrh UI) potřebovala třetinu celkového času stráveného na praxi.

Literatura

[1] Odbor strategického rozvoje statutárního města Ostravy. *FAKTOGRAFICKÉ LISTY OSTRAVA 2018*. [online]. [cit. 12. dubna 2019].

Dostupné z:
`https://www.ostrava.cz/cs/podnikatel-investor/ekonomicky-profil-mesta/ostrava-v-cislech/univerzity/univerzity-v-ostrave/FL_2018_CZ_final.pdf`

[2] IT-Slovnik.cz team. *IT SLOVNÍK*. [online]. [cit. 12. dubna 2019].

Dostupné z: `https://it-slovnik.cz/pojem/dashboard`

[3] THOMAS, Anthony. *When to Use White Text on a Dark Background*. [online]. [cit. 12. dubna 2019].

Dostupné z: `https://uxmovement.com/content/when-to-use-white-text-on-a-dark-background/`

[4] Facebook Inc. *Tutorial: Intro to React*. [online]. [cit. 12. dubna 2019].

Dostupné z: `https://reactjs.org/tutorial/tutorial.html`

[5] GISTIA LABS Engineering Team. *A Beginners Guide to Redux*. [online]. [cit. 12. dubna 2019].

Dostupné z: `https://www.gistia.com/beginners-guide-redux/`

[6] ABRAMOV, Dan and the Redux documentation authors. *Three Principles*. [online]. [cit. 12. dubna 2019].

Dostupné z: `https://redux.js.org/introduction/three-principles`

[7] ABRAMOV, Dan. *Glossary*. [online]. [cit. 12. dubna 2019].

Dostupné z: `https://redux.js.org/glossary#action`

[8] ABRAMOV, Dan. *Solution for simple action creation without string constants and less magic*. [online]. [cit. 12. dubna 2019].

Dostupné z:
`https://github.com/reduxjs/redux/issues/628#issuecomment-137547668`